College of Engineering, Forestry, and Natural Sciences

Group:	James Beasley, Charles Beck, Charles Duso, Alexander Grzesiak, Erik Strauss
Project Title:	Boston University - Microfluid Experimentation Data Generator
Deliverable:	D.5 Internal Design
Course:	CS386 – Spring 2017
Instructor:	Professor Gerosa
Github:	https://github.com/TheAwesomeEgg/CS386ProjectGroup1.git

1 Introduction

The purpose of this document is to examine the macro structure of the system we are designing as well as the components of the system and how they interact. This document will be heavily visual-based with textual descriptions to support each visual. As a reminder, we are developing a hardware instruction generator for the microfluidic experimentation group at Boston University.

2 System Architecture

In this section, we consider the system-wide architecture. We will first discuss the architecture we have implemented in detail and then explain the styles and patterns we employed to achieve our architecture. Visuals will accompany textual descriptions where necessary.

2.1 Architecture Description
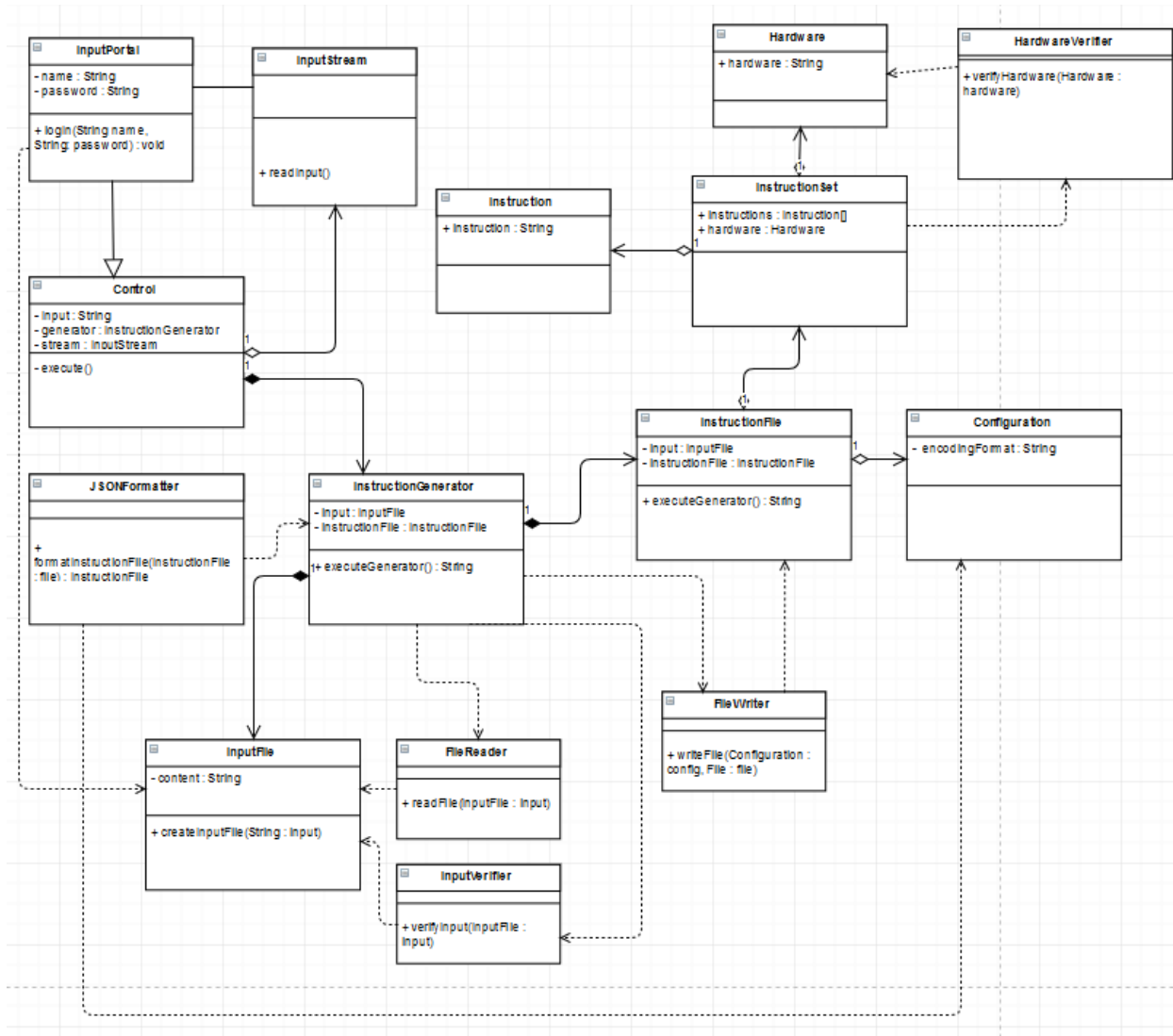
….

2.2 Architecture Style and Design Patterns

….

3 System Descriptions

In this section, we consider the interplay of the components that makeup the system architecture. We have included several diagrams that serve to demonstrate functionality, implementation and execution of the system.

3.1 Class Diagram

We have discussed the class diagram for the system in previous documents, but have revised the document to account for variations in features and to better describe pre-existing components. Listed below is the revised class diagram for the system.

InputPortal
- name : String
- password : String
+ login(String name, String password) : void

InputStream
+ readInput()

Hardware
+ hardware : String

HardwareVerifier
+ verifyHardware(Hardware : hardware)

Instruction
+ Instruction : String

InstructionSet
+ instructions : Instruction[]
+ hardware : Hardware

Control
- input : String
- generator : InstructionGenerator
- stream : InoutStream
- execute()

InstructionFile
- input : InputFile
- instructionFile : instructionFile
+ executeGenerator() : String

Configuration
- encodingFormat : String

JSONFormatter
+ formatInstructionFile(InstructionFile : file) : InstructionFile

InstructionGenerator
- input : inputFile
- instructionFile : InstructionFile
1+ executeGenerator() : String

FileWriter
+ writeFile(Configuration : config, File : file)

InputFile
- content : String
+ createInputFile(String : Input)

FileReader
+ readFile(InputFile : Input)

InputVerifier
+ verifyInput(inputFile : Input)

At a glance, it easy to discern that the structure is tightly coupled. This is intended and is a design trade-off for our creation of a monolithic architecture. Tight coupling is often seen as taboo in large systems because of the extra work associated with maintenance and new features, but our project is discretely defined and will not be updated to accommodate new functionality or scalability. That being said, coupling will not be negatively impactful towards our system.

3.2 Sequence Diagram
We will now consider the flow of logic within our system. The logic for our system is relatively simple as the user's need for system is quite explicit. To describe the flow of logic in the system, we have listed the sequence diagram below.
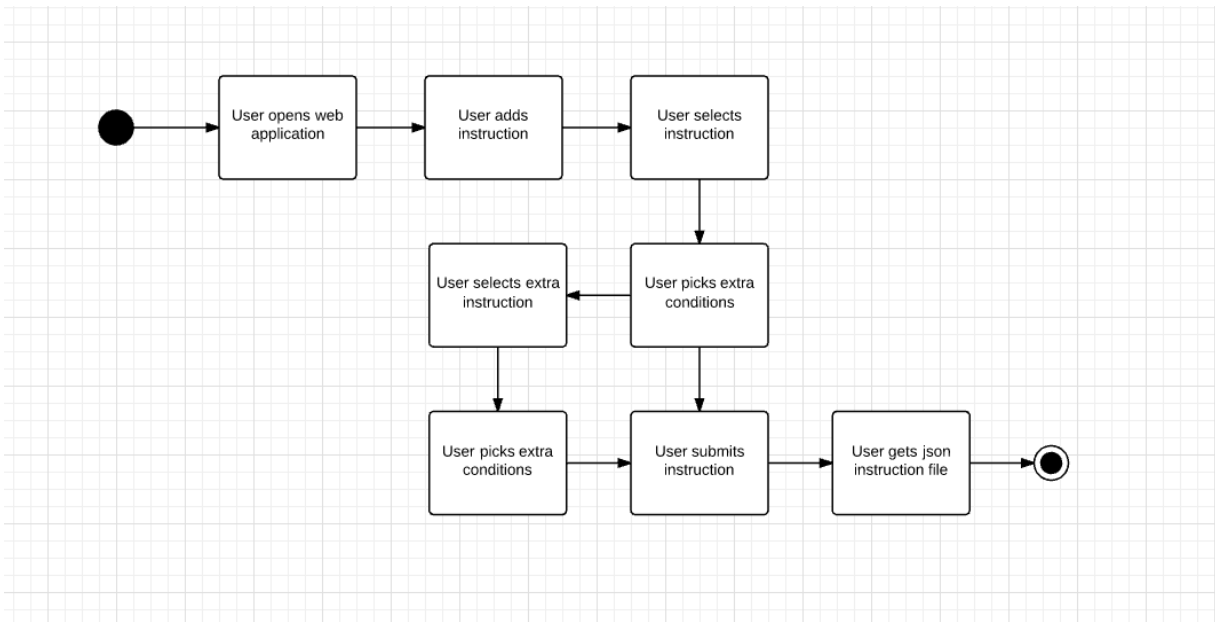
3.3 Object Diagram

….

4 Conclusion

This concludes the internal design description of the system we are designing. We are satisfied with the current design, as all requirements are satisfied; but, are prepared to make changes as needed.

Group Participation

Listed below is a table containing the group participation weights for each team member.

Team Member	Participation
James Beasley	
Charles Beck	
Charles Duso	
Alexander Grzesiak	
Erik Strauss	

Table 1: Group Participation Weights